try/except/else/finally에서 각 블록의 장점을 이용하자

파이썬에는 예외 처리 과정에서 동작을 넣을 수 있는 네 번의 구분되는 시점이 있다. try, except, else, finally 블록 기능으로 각 시점을 처리한다. 각 블록은 복합문에서 독자적인 목적이 있으며, 이 블록들을 다양하게 조합하면 유용하다

finally 블록

예외를 전달하고 싶지만, 예외가 발생해도 정리 코드를 실행하고 싶을 때 try/finally를 사용하면 된다. try/finally의 일반적인 사용 예 중 하나는 파일 핸들러를 제대로 종료하는 작업이다

```
handle = open('/tmp/random_data.txt') # IOError가 일어날 수 있음
try:
    data = handle.read() # UnicodeDecodeError가 일어날 수 있음
finally:
    handle.close()          # try: 이후에 항상 실행됨
```

read 메서드에서 발생한 예외는 항상 호출 코드까지 전달되며, handle의 close 메서드 또한 finally 블록에서 실행되는 것이 보장된다. 파일이 없을 때 일어나는 IOError처럼, 파일을 열 때 일어나는 예외는 finally 블록에서 처리하지 않아야 하므로 try 블록 앞에서 open을 호출해야 한다.

else 블록

코드에서 어떤 예외를 처리하고 어떤 예외를 전달할지를 명확하게 하려면 try/except/else를 사용해야 한다. try 블록이 예외를 일으키지 않으면 else 블록이 실행된다. else 블록을 사용하면 try 블록의 코드를 최소로 줄이고 가독성을 높일 수 있다. 예를 들어 문자열에서 JSON 딕셔너리 데이터를 로드하여 그 안에 든 키의 값을 반환한다고 하자.

```
def load_json_key(data, key):
try:
    result_dict = json.loads(data)      # ValueError가 일어날 수 있음
except ValueError as e:
    raise KeyError from e
else:
    return result_dict[key]              # KeyError가 일어날 수 있음
```

데이터가 올바른 JSON이 아니라면 json.loads로 디코드할 때 ValueError가 일어난다. 이 예외(exception)는 except 블록에서 발견되어 처리된다. 디코딩이 성공하면 else 블록에서 키를 찾는다. 키를 찾을 때 어떤 예외가 일어나면 그 예외는 try 블록 밖에 있으므로 호출 코드까지 전달된다. else 절은 try/except 다음에 나오는 처리를 시각적으로 except 블록과 구분해준다. 그래서 예외 전달 행위를 명확하게 한다.'

모두 함께 사용하기

복합문 하나로 모든 것을 처리하고 싶다면 try/except/else/finally를 사용하면 된다. 예를 들어 파일에서 수행할 작업 설명을 읽고 처리한 후 즉석에서 파일을 업데이트한다고 하자. 여기서 try 블록은 파일을 읽고 처리하는 데 사용한다. except 블록은 try 블록에서 일어난 예외를 처리하는 데 사용한다. else 블록은 파일을 즉석에서 업데이트하고 이와 관련한 예외가 전달되게 하는 데 사용한다. finally 블록은 파일

핸들을 정리하는 데 사용한다.

```
UNDEFINED = object()

def divide_json(path):
    handle = open(path, 'r+')       # IOError가 일어날 수 있음
    try:
        data = handle.read()    # UnicodeDecodeError가 일어날 수 있음
        op = json.loads(data)  # ValueError가 일어날 수 있음
        value = (
        op['numerator'] /
        op['denominator']) # ZeroDivisionError가 일어날 수 있음
    except ZeroDivisionError as e:
        return UNDEFINED
    else:
        op['result'] = value
        result = json.dumps(op)
        handle.seek(0)
        handle.write(result)    # IOError가 일어날 수 있음
        return value
    finally:
        handle.close()          # 항상 실행함
```

이 레이아웃은 모든 블록이 직관적인 방식으로 엮여서 동작하므로 특히 유용하다. 예를 들어 결과 데이터를 재작성하는 동안에 else 블록에서 예외가 일어나도 finally 블록은 여전히 실행되어 파일 핸들을 닫는다.

핵심 정리

- try/finally 복합문을 이용하면 try 블록에서 예외 발생 여부와 상관없이 정리 코드를 실행할 수 있다.
- else 블록은 try 블록에 있는 코드의 양을 최소로 줄이는 데 도움을 주며 try/except 블록과 성공한 경우에 실행할 코드를 시각적으로 구분해준다.
- else 블록은 try 블록의 코드가 성공적으로 실행된 후 finally 블록에서 공통 정리 코드를 실행하기 전에 추가 작업을 하는 데 사용할 수 있다.

다음에서 발췌: '파이썬 코딩의 기술.' Apple Books.